An Evolutionary Artificial Neural Networks Approach for Breast Cancer Diagnosis

Hussein A. Abbass
School of Computer Science,
University of New South Wales,
Australian Defence Force Academy Campus,
Northcott Drive, Canberra ACT, 2600, Australia,
h.abbass@adfa.edu.au

Abstract

This paper presents an *evolutionary artificial neural network* approach based on the *pareto differential evolution* algorithm augmented with local search for the prediction of breast cancer. The approach is named *memetic pareto artificial neural network* (MPANN). *artificial neural networks* could be used to improve the work of medical practitioners in the diagnosis of breast cancer. Their abilities to approximate nonlinear functions and capture complex relationships in the data are instrumental abilities which could support the medical domain. We compare our results against an evolutionary programming approach and standard backpropagation and we show experimentally that MPANN has better generalization and much lower computational cost.

Keywords Pareto optimization, differential evolution, artificial neural networks, breast cancer.

1 Introduction

The economic and social values of *Breast Cancer Diagnosis* (BCD) are very high. As a result, the problem has attracted many researchers in the area of computational intelligence recently

[6, 8, 10, 22, 26, 32, 33, 34]. Because of the importance of achieving highly accurate classification, *Artificial Neural Networks* (ANNs) are among the most common methods for BCD.

Research in the area of using ANNs for medical purposes – more specifically BCD – [6, 8, 10, 22, 26, 32, 34] has been at the center of attention for several years. Unfortunately, to our present knowledge, none of this type of research was able to enter the clinic either in terms of routine use or to replace the radiologist. This could be ascribed to a number of factors. The first problem was the misconception that the machines could replace the job of the radiologist. It is imperative to clarify that ANN does not aim to replace the radiologist, but rather support him/her. A second reason is the human factor, as people still disbelieve and doubt the abilities of the machine, especially when it comes to a medical diagnosis. Nevertheless, one needs to notice that it is rarely the case when a patient asks the doctor "why do you think I do not have cancer?". Actually, we are more inclined to ask why we have the disease more than asking why we do not have it.

Within this framework, ANN research aims to provide a filter that distinguishes the cases which do not have cancer; therefore reducing the cost of medication and helping the doctors to focus on real patients. The objective of ANNs is to support doctors and not to replace them. In addition, ANNs can provide a valuable tool that could minimize the disagreement and inconsistencies in mammographic interpretation. If we know that there is an excellent doctor who is accurate in his/her decisions, we build a concept of trust and we rarely need a justification from the doctor for his/her decisions. In the same line of thinking, for the use of ANNs in the clinic, what is needed by ANNs is to ensure their accuracy and reliability as tools for BCD. We need to form a model of trust between human and machines based on the accurate performance of the machine. Thus, if the patient is diagnosed by the ANN that he/she does not have cancer over a number of times and all the decisions were confirmed by a human doctor, a model of trust will emerge between the patient and the machine. As a result, ANNs' application in the medical domain in general will be extensive.

Previous research in this arena has been undertaken by various researchers. Wu *et al.* [34] used an ANN to learn from 133 instances containing each 43 mammographic features rated between 0-10 by a mammographer. The ANN was trained using the *backpropagation algorithm* (BP) [25]

using 10 hidden nodes and a single output node was trained to produce 1 for malignancy and 0 for benign. The performance of the ANNs was found to be competitive to the domain expert and after a considerable amount of feature selection, the performance of the ANNs improved and significantly outperformed the domain expert.

Another use of BP was undertaken by Floyd *et al.* [6] who used eight input parameters; these are mass size and margin, asymmetric density, architectural distortion, calcification number, morphology, density and distribution. After extensive experiments with BP over their limited dataset of 260 cases, they achieved a classification accuracy of 50%. However, as suggested by Fogel *et al.* [8], these results were questionable as they impose significant bias in the data. Also, since they did not employ a test set, their model may have over–fitted the data.

Wilding *et al.* [32] has also suggested the usage of BP where they followed a similar BP approach to the previous two references but with different input sets derived from a group of blood tests. However, with 104 instances and 10 inputs, it seems that their ANN failed to perform well.

BP suffers the disadvantage of being easily trapped in a local minimum. Therefore, Fogel *et al.* [8] used an evolutionary programming approach to train the ANN to overcome the disadvantage of BP. They used a population of 500 networks and evolved the population for 400 generations, therefore generate around 20,000 potential networks. The approach was tested on the Wisconsin dataset that we are being using in this paper, which is obtainable by anonymous ftp from ice.uci.edu [3]. They managed to achieve a significant result with 98% of the test cases correctly classified. Apart from their few trials (only 5 repeats for each experiment) and the dependence of their approach on a predefined network architecture, their approach performed considerably well, compared to the previous studies.

In another study, Setiono [26] used his rule extraction from ANNs algorithm [28, 29] to extract useful rules that can predict breast cancer from the Wisconsin dataset. He needed first to train an ANN using BP and achieved an accuracy level on the test data of approximately 94%. After applying his rule extraction technique, the accuracy of the extracted rule set did not change. In a

more recent work, Setiono [27] used feature selection before training the ANN. The new rule sets had an average accuracy of more than 96%. This is an improvement when compared to the initial results. It is also comparable to the results of Fogel *et al.* [8].

Furundzic *et al.* [10] presented another BP–ANN attempt where they used 47 input features and after the use of some heuristics to determine the number of hidden units, they used 5 hidden units. With 200 instances and after significant amount of feature selection, they reduced the number of input features to 29 while maintaining the same classification accuracy of 95%.

More recently, Pendharkar *et al.* [22] presented a comparison between the data envelopment analysis and ANN. They found that the ANN approach was significantly better than the data envelopment analysis approach with around 25% improvement in the classification accuracy.

In sum, all previous methods, except Fogel *et al.* [8, 9], depended on the conventional BP algorithm which can easily be trapped in a local minimum and requires extensive computational time to find the best number of hidden units. The approach of Fogel *et al.* [8, 9] presented a successful attempt to use evolutionary computations for solving the problem. Although the approach achieved better predictive accuracy than the others, it suffered from its high computational cost and dependance on either knowing the right number of hidden units in advance or experimenting with a number of different networks to find the best number of hidden units. Therefore, the results of previous studies indicated a need for an approach that is less expensive, can find the right number of hidden units without so much interference from the user, and is as accurate as that of Fogel *et al.* [8, 9].

In this light, the objective of this paper is to present a *Memetic* (*ie. evolutionary algorithms* (EAs) augmented with local search [20]) *pareto artificial neural networks* (MPANNs) algorithm for BCD. This approach is found to be as accurate as the method of Fogel *et al.* but with much lower computational cost. The rest of the paper is organized as follows: In Section 2, background materials are covered followed by an explanation of the method in Section 3. Results are discussed in Section 4 and conclusions are drawn in Section 5.

2 Background Materials

Evolutionary artificial neural networks (EANNs) have been a key research area for the last decade. On the one hand, methods and techniques have been developed to find better approaches for evolving *artificial neural networks* and more precisely, for the sake of our paper, multi–layer feed–forward ANNs. On the other hand, finding a good ANNs' architecture has been a debatable issue as well in the field of ANNs. Methods for network growing (such as Cascade Correlation [5]) and for network pruning (such as Optimal Brain Damage [16]) have been used to overcome the long process for determining a good network architecture. However, all these methods still suffer from their slow convergence and long training time. In addition, they are based on gradient–based techniques and therefore can easily stuck in a local minimum. EANNs provide a more successful platform for optimizing both the network performance and architecture simultaneously. Unfortunately, most of the research undertaken in the EANN literature does not emphasize the trade–off between the architecture and the generalization ability of the network. A network with more hidden units may perform better on the training set, but may not generalize well on the test set. This trade–off is a well known problem in *optimization* known as the Multi-objective Optimization Problem (MOP).

MOP is an active area of research in the field of *operations research* for the last five decades, since the introduction of the weighted sum method by Kuhn and Tucker [15]. Research into MOP flourished again in the last decade with the massive use of evolutionary approaches in single–objective optimization. Applying EA to MOPs provided the natural ingredients for solving complex MOPs. These approaches will be briefly outlined in the following section.

With the trade–off between the network architecture – taken in this paper to be the number of hidden units – and the generalization error, the EANN problem is in effect an MOP. It is, therefore, natural to raise the question of why not applying a multi–objective approach to EANN.

In the following section, we introduce necessary background materials for multi-objective opti-

mization, ANNs, EANNs, and the *pareto-differential evolution* (PDE) approach.

2.1 Multi-objective Optimization

Consider a MOP model as presented below:-

$$\text{Optimize } F(\vec{x})$$
$$\text{subject to: } \Omega = \{\vec{x} \in R^n | G(\vec{x}) \leq 0\}$$

Where $\vec{x}$ is a vector of decision variables (x_1, ..., x_n) and $F(\vec{x})$ is a vector of objective functions ($f_1(\vec{x})$, ..., $f_K(\vec{x})$). Here $f_1(\vec{x}), \ldots, f_K(\vec{x})$ are functions on R^n and Ω is a nonempty set in R^n. The vector $G(\vec{x})$ represents a set of constraints.

In MOPs, the aim is to find the optimal solution $\vec{x}^* \in \Omega$ which optimize $F(\vec{x})$. Each objective function, $f_i(\vec{x})$, is either maximization or minimization. Without any loss of generality, we assume that all objectives are to be minimized for clarity purposes. We may note that any maximization objective can be transformed to a minimization one by multiplying the former by -1.

To define the concept of non-dominated solutions in MOPs, we need to define two operators, $\gneqq$ and $\precnsim$ and then assume two vectors, $\vec{x}$ and $\vec{y}$. We define the first operator as $\vec{x} \gneqq \vec{y}$ iff $\exists\ x_i \in \vec{x}$ and $y_i \in \vec{y}$ such that $x_i \neq y_i$. And, $\vec{x} \precnsim \vec{y}$ iff $\forall\ x_i \in \vec{x}$ and $y_i \in \vec{y}, x_i \leq y_i$, and $\vec{x} \gneqq \vec{y}$. The operators $\gneqq$ and $\precnsim$ can be seen as the "not equal to" and "less than or equal to" operators respectively, between two vectors. We can now define the concepts of local and global optimality in MOPs.

Definition 1: Neighborhood or open ball The open ball (*ie.* a neighborhood centered on $\vec{x}^*$ and defined by the Euclidean distance) $B_\delta(\vec{x}^*) = \{\vec{x} \in R^n |\ ||\vec{x} - \vec{x}^*|| < \delta\}$.

Definition 2: Local efficient (non-inferior/ pareto-optimal) solution A vector $\vec{x}^* \in \Omega$ is said to be a local efficient solution of MOP iff $\nexists\ \vec{x} \in (B_\delta(\vec{x}^*) \cap \Omega)$ such that $F(\vec{x}) \precnsim F(\vec{x}^*)$ for some positive δ.

Definition 3: Global efficient (non-inferior/ pareto-optimal) solution A vector $\vec{x}^* \in \Omega$ is said to be a global efficient solution of MOP iff $\nexists\, \vec{x} \in \Omega$ such that $F(\vec{x}) \precnsim F(\vec{x}^*)$.

Definition 4: Local non-dominated solution A vector $\vec{y}^* \in F(\vec{x})$ is said to be local non-dominated solution of MOP iff its projection onto the decision space, $\vec{x}^*$, is a local efficient solution of MOP.

Definition 5: Global non-dominated solution A vector $\vec{y}^* \in F(\vec{x})$ is said to be global non-dominated solution of MOP iff its projection onto the decision space, $\vec{x}^*$, is a global efficient solution of MOP.

In this paper, the term "non-dominated solution" is used as a shortcut for the term "local non-dominated solution".

2.2 Artificial Neural Networks

We may define an ANN by a graph: $G(N, A, \psi)$, where N is a set of neurons (also called nodes), A denotes the connections (also called arcs or synapses) between the neurons, and ψ represents the learning rule whereby neurons are able to adjust the strengths of their interconnections. A neuron receives its inputs (also called activation) from an external source or from other neurons in the network. It then undertakes some processing on this input and sends the result as an output. The underlying function of a neuron is called the activation function. The activation, a, is calculated as a weighted sum of the inputs to the node in addition to a constant value called the bias. The bias can be easily augmented to the input set and considered as another input. From herein, the following notations will be used for a single hidden layer MLP:

- I and H are the number of input and hidden units respectively.
- $\mathbf{X}^p \in \mathbf{X} = (x_1^p, x_2^p, \ldots, x_I^p), p = 1, \ldots, P$, is the pth pattern in the input feature space $\mathbf{X}$ of dimension I, and P is the total number of patterns.
- Without any loss of generality, $\mathbf{Y}_o^p \in \mathbf{Y_o}$ is the corresponding scalar of pattern $\mathbf{X}^p$ in the hypothesis space $\mathbf{Y_o}$.

- w_{ih} and w_{ho}, are the weights connecting input unit i, $i = 1, \ldots, I$, to hidden unit h, $h = 1 \ldots H$, and hidden unit h to the output unit o (where o is assumed to be 1 in this paper) respectively.

- $\Theta_h(\mathbf{X}^p) = \sigma(a_h)$; $a_h = \sum_{i=0}^{I} w_{ih} x_i^p$, $h = 1, \ldots, H$, is the h^{th} hidden unit's output corresponding to the input pattern $\mathbf{X}^p$, where a_h is the activation of hidden unit h, and $\sigma(.)$ is the activation function that is taken in this paper to be the logistic function $\sigma(z) = \frac{1}{1+e^{-Dz}}$, with D the function's sharpness or steepness and is taken to be 1 unless it is mentioned otherwise.

- $\hat{Y}_o^p = \sigma(a_o)$; $a_o = \sum_{h=0}^{H} w_{ho} \Theta_h(\mathbf{X}^p)$ is the network output and a_o is the activation of output unit o corresponding to the input pattern $\mathbf{X}^p$.

MLPs are in essence non-parametric regression methods which approximate underlying functionality in data by minimising a risk function. The data are presented to the network and the risk function is approximated empirically R_{emp} by summing over all data instances as follows:

$$R_{emp}(\alpha) = \sum_{p=1}^{P} (Y_o^p - \hat{Y}_o^p)^2 \tag{1}$$

The common loss functions used for training an ANN are the *quadratic* error function and the *entropy*. If the loss function is quadratic, the task is to minimize the mean (expected) square error (MSE) between the actual observed or target values and the corresponding values predicted by the network (Equation 1). The backpropagation algorithm, developed initially by Werbos [31] and then independently by Rumelhart group [25], is commonly used for training the network. BP deploys the gradient of the empirical risk function to alter the parameter set α until the risk is minimum. BP in its simple form uses a single parameter, η representing the learning rate. For a complete description for the derivations of this algorithm, see for example [11]. The algorithm can be described in the following steps:

1. Until termination conditions are satisfied, do

   (a) for each input-output pairs, $(\mathbf{X}^p, Y_o^p)$, in the training set, apply the following steps

      i. Inject the input pattern $\mathbf{X}^p$ into the network

ii. Calculate the output, $\Theta_h(\mathbf{X}^p)$, for each hidden unit h.

iii. Calculate the output, $\hat{Y}_o^{\,p}$, for each output unit o.

iv. for the output unit o, calculate $r_o = \hat{Y}_o^p(1-\hat{Y}_o^p)(Y_o^p - \hat{Y}_o^p)$ where r_o is the rate of change in the error of the output unit o.

v. for each hidden unit h, $r_h = \Theta_h^p(1-\Theta_h^p)w_{ho}r_o$ where r_h is the rate of change in the error of hidden unit h.

vi. update each weight in the network using the learning rate η as follows:

$$w_{ih} \leftarrow w_{ih} + \Delta w_{ih}, \qquad \Delta w_{ih} = \eta r_j a_{ih} \tag{2}$$

$$w_{ho} \leftarrow w_{ho} + \Delta w_{ho}, \qquad \Delta w_{ho} = \eta r_k a_{ho} \tag{3}$$

2.3 Evolutionary Artificial Neural Networks

Over the last two decades, research into EANN has witnessed a flourishing period [36, 37]. Yao [38] presents a thorough review to the field with over 300 references just in the area of EANN. This may indicate that there is an extensive need for finding better ways to evolve ANN.

A major advantage to the evolutionary approach over traditional learning algorithms such as *backpropagation* (BP) is the ability to escape a local optimum. More advantages include robustness and ability to adopt in a changing environment. In the literature, research into EANN has been taking one of three approaches; evolving the weights of the network, evolving the architecture, or evolving both simultaneously.

The EANN approach uses either binary representation to evolve the weight matrix [12, 13] or real [7, 8, 9, 18, 19, 23]. There is not an obvious advantage of binary encoding in EANN over the real. However, with real encoding, there are more advantages including compact and natural representation.

The key problem (other than being trapped in a local minimum) with BP and other traditional training algorithms is the choice of a correct architecture (number of hidden nodes and connections). This problem has been tackled by the evolutionary approach in many studies [4, 14, 17, 21, 24, 39,

40, 41]. In some of these studies, weights and architectures are evolved simultaneously.

The major disadvantage to the EANN approach is it is computationally expensive, as the evolutionary approach is normally slow. To overcome the slow convergence of the evolutionary approach to ANN, hybrid techniques were used to speed up the convergence by augmenting evolutionary algorithms with a local search technique (*ie.* memetic approach), such as BP [35].

In general, most of the previous approaches evolve either the weights or the networks architecture. Even those that evolve both simultaneously, they do it in two stages. None of these approaches actually considers the dependency between the network architecture and weights. The objective of the proposed method in this paper is to train and determine the number of hidden units in the network simultaneously. Thus, in the following section, the proposed method is introduced.

2.4 Pareto-Differential Evolution

Abbass et al. [1] described the Pareto-frontier Differential Evolution (PDE) algorithm for vector optimization problems. The algorithm is an adaptation of the original *Differential evolution* (DE) introduced by Storn and Price [30] for optimization problems over continuous domains. The PDE method outperformed all previous methods on five benchmark problems. PDE works as follows:

1. Create an initial population of potential solutions at random from a Gaussian distribution $N(0.5, 0.15)$.
2. Repeat
   (a) Delete dominated solutions
   (b) Repeat
      i. Select at random an individual as the main parent, and two individuals as supporting parents.
      ii. With some probability, called the *crossover probability*, each variable in the main parent is perturbed by adding to it a ratio, F, of the difference between the two values

of this variable in the other two supporting parents. The step-length parameter F is generated from a Gaussian distribution $N(0,1)$. At least one variable must be changed. This process represents the crossover operator in DE.

iii. If the resultant vector dominates the main parent, the resultant vector is placed into the population; otherwise it is omitted.

(c) Until the population size is maximum

3. Until termination conditions are satisfied

The algorithm works as follows. Assuming that all variables are bounded between (0,1), an initial population is generated at random from a Gaussian distribution with mean 0.5 and standard deviation 0.15. All dominated solutions are removed from the population. The remaining non-dominated solutions are retained for reproduction. Three parents are selected at random (one as a main and as a trial solution and the other two as supporting parents). A child is generated from the three parents and is placed into the population if it dominates the main parent; otherwise a new selection process takes place. This process continues until the population is completed.

3 The MPANN Algorithm

3.1 Representation

In deciding on an appropriate representation, we tried to choose a representation that can be used for other architectures without further modifications. Our chromosome is a class that contains one matrix Ω and one vector ρ. The matrix Ω is of dimension $(I+O) \times (H+O)$. Each element $\omega_{ij} \in \Omega$, is the weight connecting unit i with unit j, where $i = 0, \ldots, (I-1)$ is the input unit i, $i = I, \ldots, (I+O-1)$ is the output unit $(i-I)$, $j = 0, \ldots, (H-1)$ is the hidden unit j, and $j = H, \ldots, (H+O-1)$ is the output unit $(j-H)$. This representation has the following two characteristics that we are not using in the current version but can easily be incorporated in the algorithm in future work:

1. It allows direct connection from each input to each output units (we allow more than a single output unit in our representation).

2. It allows recurrent connections between the output units and themselves.

The vector ρ is of dimension H, where $\rho_h \in \rho$ is a binary value used to indicate if hidden unit h exists in the network or not; that is, it works as a switch to turn a hidden unit on or off. The sum, $\sum_{h=0}^{H} \rho_h$, represents the actual number of hidden units in a network, where H is the maximum number of hidden units. This representation allows simultaneous training of the weights in the network and selecting a subset of hidden units.

3.2 Methods

As the name indicates in our proposed method, we have a multi–objective problem with two objectives; one is to minimize the error and the other is to minimize the number of hidden units. The pareto–frontier of the tradeoff between the two objectives will have a set of networks with different number of hidden units (note the definition of pareto-optimal solutions). However, sometimes the algorithm will return two pareto-networks with the same number of hidden units. This will only take place when the actual number of pareto-optimal solutions in the population is less than 3. Because of the condition in DE of having at least 3 parents in each generation, if there are less than three parents, the pareto optimal solutions are removed from the population and the population is re-evaluated. To exemplify this, if we assume that we have only a single pareto optimal solution in the population, we need another 2. The process simply starts by removing the pareto optimal solution from the population and finding the pareto optimal solutions in the remainder of the population. Those solutions dominating the rest of the population are added to the pareto set until the number of pareto solutions in the set is 3.

Our proposed method amalgamates the original PDE [1] algorithm with local search (*ie.* BP) to form the memetic approach. In initial investigations, the algorithm was quite slow and the use of local search improved its performance. MPANN consists of the following steps:

1. Create a random initial population of potential solutions. The elements of the weight matrix Ω are assigned random values according to a Gaussian distribution $N(0,1)$. The elements of the binary vector ρ are assigned the value 1 with probability 0.5 based on a randomly generated number according to a uniform distribution between $[0,1]$; otherwise 0.

2. Repeat

   (a) Evaluate the individuals in the population and label those who are non-dominated.

   (b) If the number of non-dominated individuals is less than 3 repeat the following until the number of non-dominated individuals is greater than or equal to 3:

      i. Find a non-dominated solution among those who are not labelled.

      ii. Label the solution as non-dominated.

   (c) Delete all dominated solutions from the population.

   (d) Mark 20% of the training set as a validation set for BP.

   (e) Repeat

      i. Select at random an individual as the main parent α_1, and two individuals, α_2, α_3 as supporting parents.

      ii. **Crossover:** with some probability $Uniform(0,1)$, do

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{\alpha_1} + N(0,1)(\omega_{ih}^{\alpha_2} - \omega_{ih}^{\alpha_3}) \tag{4}$$

$$\rho_h^{child} \leftarrow \begin{cases} 1 & if(\rho_h^{\alpha_1} + N(0,1)(\rho_h^{\alpha_2} - \rho_h^{\alpha_3})) \geq 0.5 \\ 0 & otherwise \end{cases} \tag{5}$$

      otherwise

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{\alpha_1} \tag{6}$$

$$\rho_h^{child} \leftarrow \rho_h^{\alpha_1} \tag{7}$$

      and with some probability $Uniform(0,1)$, do

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{\alpha_1} + N(0,1)(\omega_{ho}^{\alpha_2} - \omega_{ho}^{\alpha_3}) \tag{8}$$

otherwise

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{\alpha_1} \tag{9}$$

where each weight in the main parent is perturbed by adding to it a ratio, $F \in N(0,1)$, of the difference between the two values of this variable in the two supporting parents. At least one variable must be changed.

iii. **Mutation:** with some probability $Uniform(0,1)$, do

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{child} + N(0, mutation\_rate) \tag{10}$$

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{child} + N(0, mutation\_rate) \tag{11}$$

$$\rho_{h}^{child} \leftarrow \begin{cases} 1 & if \rho_{h}^{child} = 0 \\ 0 & otherwise \end{cases} \tag{12}$$

iv. Apply BP to the child.

v. If the child dominates the main parent, place it into the population.

(f) Until the population size is M

3. Until termination conditions are satisfied, go to 2 above.

One may note that before each generation starts, 20% of the instances in the training set are marked as a validation set for the use of BP; that is, BP will use 80% of the original training set for training and 20% for validation. Also, the termination condition in our experiments occurs when the maximum number of epochs is reached; where one epoch is equivalent to one pass through the training set. Therefore, one iteration of BP is equivalent to one epoch since 80% of the training set is used for training and the other 20% for validation; that is, one complete pass through the original training set. The cross–validation set is used to select the best generalized network during BP training.

Since some hidden units are inactive, if these hidden units are disregarded during the application of BP, the weights on their associated connections will not change. Hence, these connections may

be disadvantaged in later stages; therefore, we need to alter the way BP is used. In this case, the error on the output level is calculated using only those active hidden units, but when propagated back into the network, it is distributed among all hidden units.

The pareto approach helps to determine the number of hidden units without any interference from the user. An alternative method could be formulating the problem using a single objective by taking a weighted sum of the two objectives. However, this is not an efficient way to solve the problem. First, the weighted sum method could only generate a pareto solution at a time. Second, it assumes convexity of the pareto frontier. Third, we cannot add the error to the number of hidden units because of the different dimensionality of the two objectives. Last but not least, even if we can unify the dimensionality, the question of determining the appropriate values for the weights still remains; that is, we will need to run the algorithm a number of times with different weights which in effect is equivalent to running a conventional evolutionary approach while varying the number of hidden units. In sum, the multiobjective approach provides flexibility and reduces the user's load. In the following section, we will outline the performance of MPANN on the breast cancer dataset.

4 Diagnosis of Breast Cancer with MPANN

4.1 The Breast Cancer Dataset

MPANN is used in this section to diagnose cancer based on the Breast Cancer Wisconsin dataset available by anonymous ftp from ice.uci.edu [3]. The following is a brief description of the dataset.

This dataset contains 699 patterns with 9 attributes; viz clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. The class output includes 2 classes, benign and malignant. The original dataset was obtained by Wolberg and Mangasarian [33].

4.2 Experimental Setup

To be consistent with the literature [8], we removed the sixteen instances with missing values from the dataset to construct a new dataset with 683 instances. The first 400 instances in the new dataset are chosen as the training set and the remaining 283 as the test set. Although [8] undertakes five repeats only, we decided to do 15 repeats to get more accurate figures regarding the performance of MPANN. We varied the crossover and mutation probabilities between 0 to 1 with an increment of 0.1 to have 121 different experiments and each experiment ran for 15 times. The maximum number of epochs is set to 500, the maximum number of hidden units is set to 10, the population size 12, the learning rate for BP 0.03, and the number of epochs for BP is set to 5. It is noticeable at this point that the population size needs to be larger than the maximum number of hidden units because the maximum number of pareto-solutions in this case is the maximum number of hidden units (from the definition of pareto optimality).

4.3 Results

The average test error of the selected pareto network (the one with the smallest training error) and the corresponding number of hidden units are being calculated along with the standard deviations as shown in Table 1. It is interesting to see the small standard deviations for the test error in the dataset, which indicate consistency, stability and accuracy of the method.

Table 1: The average test accuracy and standard deviations of the selected pareto network (smallest training error) for crossover 0.10 and mutation 0.10.

Figure 1: The average test error for the Breast Cancer dataset obtained by each crossover prob-

ability for each mutation probability.

Figure 2: The average number of hidden units for the Breast Cancer dataset obtained by each crossover probability for each mutation probability.

Figure 3: The average number of objective evaluations for the Breast Cancer dataset obtained by each crossover probability for each mutation probability.

In Figures 1, 2, and 3 the average test errors, number of hidden units, and number of objective evaluations corresponding to the selected network is plotted for the eleventh mutation probabilities for each of the eleventh crossover probabilities respectively. In Figure 1, the best performance occurs with small crossover and mutation probabilities. This is quite important as it entails that the building blocks in MPANN is effective; otherwise a better performance would have occurred with the maximum crossover probability. We may note here that crossover in DE is somewhat a guided mutation operator. The best result for each crossover rate is achieved with a mutation rate ranging between 0 to 0.10; except with a crossover rate 0.20 where the mutation rate of 0.20 achieved slightly better results than the corresponding mutation rate of 0.10. In general, we can conclude from Figure 1 that since crossover in DE is some sort of a guided mutation, mutation is not very important in our problem.

In summary, the best performances for the Breast Cancer dataset achieved an average accuracy

of 0.981 ± 0.005 and average number of hidden units of 4.125 ± 1.360.

4.4 Comparisons and Discussions

A summary of results is also presented in Table 2. When comparing MPANN with Fogel et al. [8], MPANN achieves slightly better performance over the test set (0.9812 as opposed to 0.9805 of Fogel at al.) but with much lower standard deviation (0.005 as compared to 0.464). In terms of computational time, we achieved a significant reduction (5100 epochs as opposed to 200,000 of Fogel at al.; that is around 3% of the computational cost needed by Fogel at al.). In terms of resources, our population is much smaller than Fogel et al. (12 as opposed to 500), which may indicate that MPANN is capable of maintaining diversity (through its use of a MOP approach) better than the evolutionary programming used by Fogel et al. approach.

Table 2: A comparison between MPANN and the others algorithms. The average test accuracy and standard deviations of the best performance achieved by each method.

A more recent study by Abbass et al. [2] presented, as part of their analysis, results for the traditional BP algorithm for training artificial neural networks. BP achieved an accuracy of 97.5 ± 1.8 with a computational cost of 10,000 epochs. Once more we can observe that MPANN has much lower computational cost (around 50% of BP) and accomplished better generalization.

We notice here that MPANN also optimized its architecture while improving its generalization ability. Therefore, in terms of the amount of computations, it is by far faster than BP and Fogel et al. methods which run for a fixed architecture. In addition, the total number of epochs required is

small compared to the corresponding number of epochs needed by BP.

5 Conclusion

In this paper, we introduced an evolutionary multi–objective approach to artificial neural networks for the breast cancer diagnosis. We showed empirically that the proposed approach has better generalization than previous approaches with much lower computational cost. However, more work is needed in evaluating the performance of the proposed method on other medical datasets.

6 Acknowledgement

The author would like to thank Xin Yao, Bob Mckay, and Ruhul Sarker for their insightful comments while discussing the initial idea with them. We would also like to show our appreciation to the four anonymous reviewers for their invaluable comments and suggestions.

References

[1] H.A. Abbass, R. Sarker, and C. Newton. A pareto differential evolution approach to vector optimisation problems. *IEEE Congress on Evolutionary Computation, IEEE Publishing, Seoul, Korea*, 2:971–978, 2001.

[2] H.A. Abbass, M. Towsey, and G.D. Finn. C-net: A method for generating non-deterministic and dynamic multivariate decision trees. *Knowledge and Information Systems*, 3:184–197, 2001.

[3] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html. *University of California, Irvine, Dept. of Information and Computer Sciences*, 1998.

[4] S. Bornholdt and D. Graudenz. General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks*, 5:327–334, 1992.

[5] S. Fahlman and C. Lebiere. The cascade correlation learning architecture. Technical Report CMU-CW-90-100, Canegie Mellon University, Pittsburgh, PA, 1990.

[6] C.E. Floyd, J.Y. Lo, A.J. Yun, D.C. Sullivan, and P.J. Kornguth. Prediction of breast cancer malignancy using an artificial neural network. *Cancer*, 74:2944–2998, 1994.

[7] D.B. Fogel, L.J. Fogel, and V.W. Porto. Evolving neural networks. *Biological Cybernetics*, 63:487–493, 1990.

[8] D.B. Fogel, E.C. Wasson, and E.M. Boughton. Evolving neural networks for detecting breast cancer. *Cancer letters*, 96(1):49–53, 1995.

[9] D.B. Fogel, E.C. Wasson, and V.W. Porto. A step toward computer-assisted mammography using evolutionary programming and neural networks. *Cancer letters*, 119(1):93, 1997.

[10] D. Furundzic, M. Djordjevic, and A.J. Bekic. Neural networks approach to early breast cancer detection. *Systems Architecture*, 44:617–633, 1998.

[11] S. Haykin. *Neural networks - a comprehensive foundation*. Printice Hall, New Jersey, USA, 2 edition, 1999.

[12] D.J. Janson and J.F. Frenzel. Application of genetic algorithms to the training of higher order neural networks. *Systems Engineering*, 2:272–276, 1992.

[13] D.J. Janson and J.F. Frenzel. Training product unit neural networks with genetic algorithms. *IEEE Expert*, 8(5):26–33, 1993.

[14] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4(4):461–476, 1990.

[15] H.W. Kuhn and A.W. Tucker. Nonlinear programming. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, 1951.

[16] Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, pages 598–605. Morgan Kauffman, San Mateo, CA, 1990.

[17] V. Maniezzo. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, 5(1):39–53, 1994.

[18] F. Menczer and D. Parisi. Evidence of hyperplanes in the genetic learning of neural networks. *Biological Cybernetics*, 66:283–289, 1992.

[19] D. Montana and L. Davis. Training feedforward artificial neural networks using genetic algorithms. pages 762–767, Morgan Kauffman, San Mateo, CA, 1989.

[20] P. Moscato. Memetic algorithms: a short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New ideas in optimization*, pages 219–234. McGraw-Hill, London, 1999.

[21] S. Oliker, M. Furst, and O. Maimon. A distributed genetic algorithm for neural network design and training. *Complex Systems*, 6(5):459–477, 1992.

[22] P.C. Pendharkar, J.A. Rodger, G.J. Yaverbaum, N. Herman, and M. Benner. Association, statistical, mathematical and neural approaches for mining breast cancer patterns. *Expert Systems with Applications*, 17:223–232, 1999.

[23] V.W. Porto, D.B. Fogel, and L.J. Fogel. Alternative neural network training methods. *IEEE Expert*, 10(3):16–22, 1995.

[24] J.C.F. Pujol and R. Poli. Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence*, 8(1):73–84, 1998.

[25] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In J.L. McClelland D.E. Rumelhart and the PDP Research Group Eds, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition., Foundations, 1*, MIT Press, Cambridge, pages 318–362, 1986.

[26] R. Setiono. Extracting rules from pruned neural networks for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 8:37–51, 1996.

[27] R. Setiono. Generating concise and accurate classification rules for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 18:205–219, 2000.

[28] R. Setiono and L. Huan. Understanding neural networks via rule extraction. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 480–487. Morgan Kaufmann, 1995.

[29] R. Setiono and L. Huan. Neurolinear: from neural networks to oblique decision rules. *Neurocomputing*, 17:1–24, 1997.

[30] R. Storn and K. Price. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, 1995.

[31] P. Werbos. *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.

[32] P. Wilding, M.A. Morgan, A.E. Grygotis, M.A. Shoffner, and E.F. Rosato. Application of backpropagation neural networks to diagnosis of breast and ovarian cancer. *Cancer Letter*, 77:145–153, 1994.

[33] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, Washington, DC, 87:9193–9196, 1990.

[34] Y.Z. Wu, M.L. Giger, K. Doi, C.J. Vyborny, R.A. Schmidt, and C.E. Metz. Artificial neural networks in mammography: application to decision making in the diagnosis of breast cancer. *Radiology*, 187:81–87, 1993.

[35] W. Yan, Z. Zhu, and R. Hu. Hybrid genetic/bp algorithm and its application for radar target classification. *Proceedings of the 1997 IEEE National Aerospace and Electronics Conference, NAECON, IEEE Press, USA*, pages 981–984, 1997.

[36] X. Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(5):203–222, 1993.

[37] X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8(4):529–567, 1993.

[38] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE, IEEE Press, USA*, 87(9):1423–1447, 1999.

[39] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Trans. on Neural Networks*, 8(3):694–713, 1997.

[40] X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(3):417–425, 1998.

[41] X. Yao and Y. Liu. Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation*, 91(1):83–90, 1998.

Table 1: The average test accuracy and standard deviations of the selected pareto network (smallest training error) for crossover 0.10 and mutation 0.10.

.

Error	0.981 ± 0.005
Number of hidden units	4.125 ± 1.360

Table 2: A comparison between MPANN and the others algorithms. The average test accuracy and standard deviations of the best performance achieved by each method.

.

Method	Error	Number of epochs
MPANN	0.981 ± 0.005	5,100
Fogel et al.	0.981 ± 0.464	200,000
Back–Prop of Abbass et al.	0.975 ± 1.800	10,000

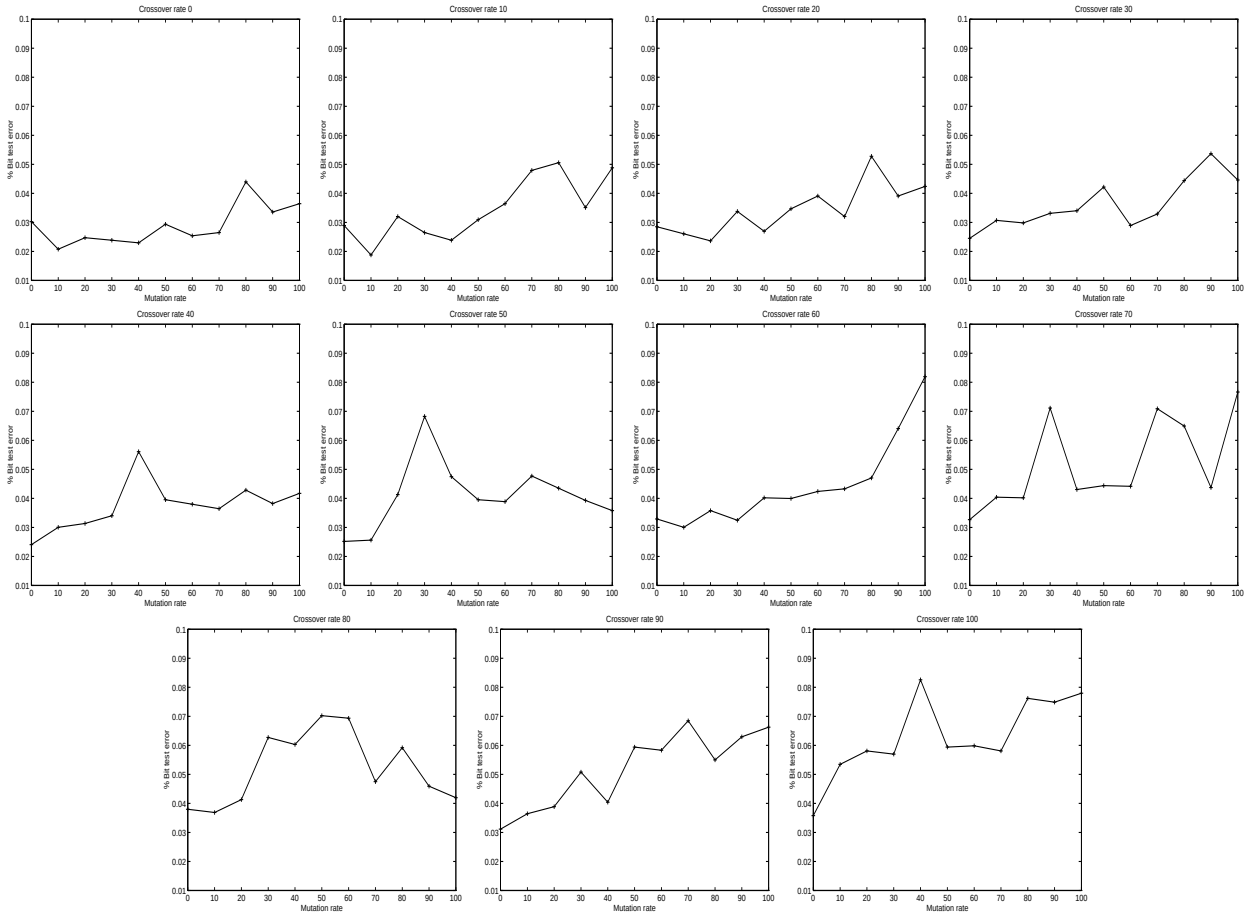

Figure 1: *The average test error for the Breast Cancer dataset obtained by each crossover probability for each mutation probability.*

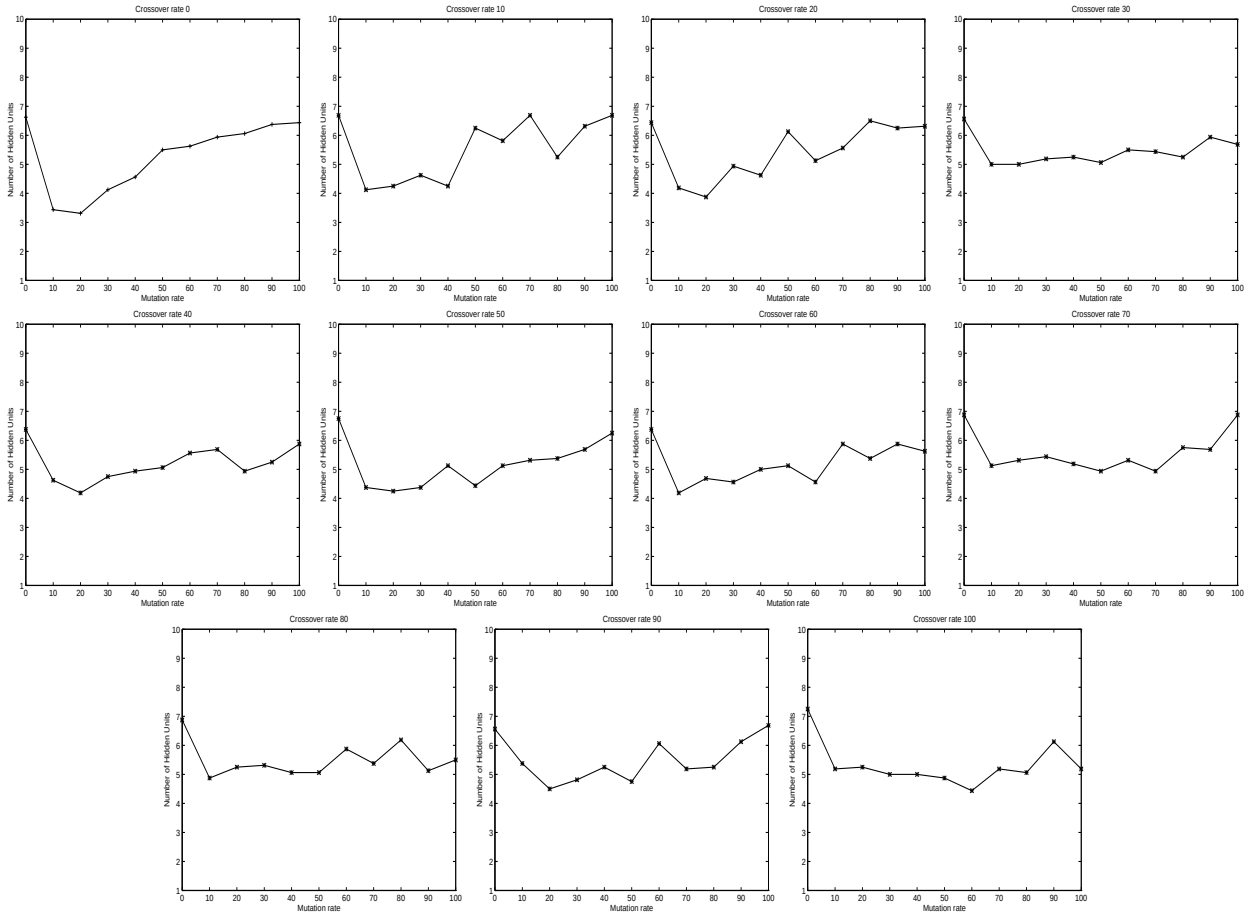

Figure 2: *The average number of hidden units for the Breast Cancer dataset obtained by each crossover probability for each mutation probability.*

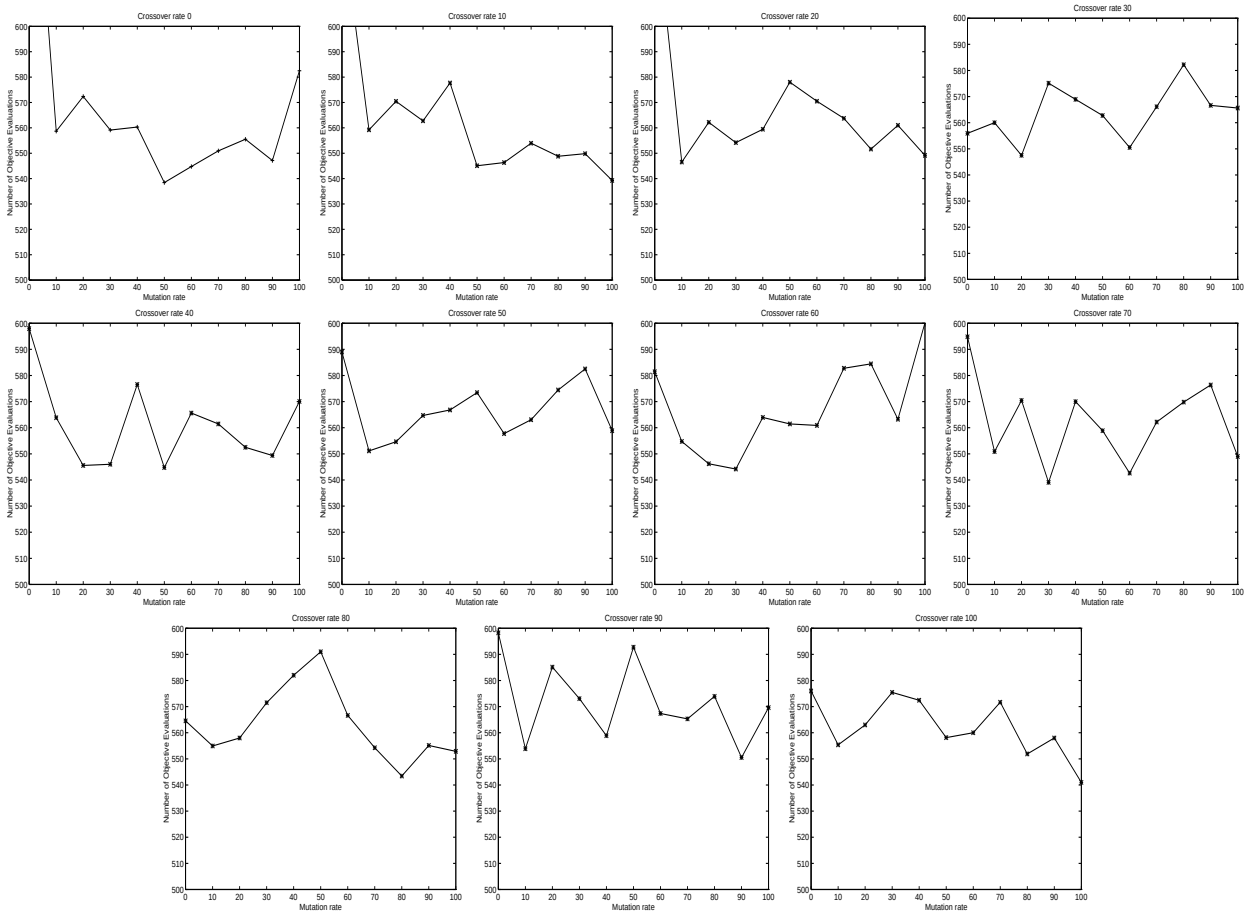

Figure 3: *The average number of objective evaluations for the Breast Cancer dataset obtained by each crossover probability for each mutation probability.*

Table 1: The average test accuracy and standard deviations of the selected pareto network (smallest training error) for crossover 0.10 and mutation 0.10.

Table 2: A comparison between MPANN and the others algorithms. The average test accuracy and standard deviations of the best performance achieved by each method.

Figure 1: The average test error for the Breast Cancer dataset obtained by each crossover probability for each mutation probability.

Figure 2: The average number of hidden units for the Breast Cancer dataset obtained by each crossover probability for each mutation probability.

Figure 3: The average number of objective evaluations for the Breast Cancer dataset obtained by each crossover probability for each mutation probability.